Sommaire du document :

1. Indentation

- ***Tabulation :*** *4 ou 8 caractères (à régler dans les préférences de l'éditeur)*

```
switch (charTest) {
case 'A':
case 'B':
        data <<= 20;
        break;
default:
        break;
}
```

- ***Nombres de niveaux d'indentation :*** *''Éviter'' plus de 3 niveaux d'indentation imbriqués. Permet de rendre le code plus lisible.*

- ***Nombres de colonnes par page :*** *~80 Colonnes Max. Permet d'améliorer la lisibilité du code.*

```
void funcTest (int nbChar, int doNothing) {

    if (condition)
        printf("Attention, si possible ne pas depasser %d caractères \
        par ligne de code",  nbChar);
    else
        printf("sinon");
}
```

- ***Structures de contrôle :*** *règles spécifiant l'indentation ainsi que les espaces à respecter pour l'écriture de structures de contrôle. Inspiré de la syntaxe K&R.*

```
do {
      // instructions multiples
      if (a == b) {
            ...
      } else if (a > b) {
            ...
      } else {
            ...
      }
      // instructions uniques
      if (a == b)
            instUnique1();
      else
            instUnique2();

} while (condition);
```

2. Dénominations

- ***Fonctions :*** *Toujours débuter par une minuscule. Pour donner un nom complexe, utiliser comme séparateur une Majuscule ou un ''_''.*

- ***Variables locales :*** *Toujours débuter par une minuscule. Les variables servant de compteur de boucle peuvent avoir un nom sans sens précis, par exemple ''i''. Sinon, toujours donner à une variable un nom permettant d'identifier clairement son rôle. Déclarez le plus souvent possible vos variables locales en début de fonction.*

- ***Variables Globales :*** *Toujours débuter par une minuscule. Toujours donner à une variable globale un nom permettant d'identifier clairement son rôle.*

```
int dataConv = 6;
 ...
Void dataAcqu_AIC23 (void) {

      int i;

      for(i = 0; i < SIZE; i++) {
            ...
      }
}
```

- ***Typedefs :*** *Toujours débuter par une Majuscule. Toujours définir un nom permettant d'identifier clairement le type des variables déclarés.*

```
HandleSpiModule a;      // Bien !
Hmod a;                 // Pas Bien !
```

- ***Macros :*** *Toujours en Majuscule. Utiliser comme séparateur un ''_'' pour les noms complexes. Pour les macros fonctions, appliquer les même règles de dénomination que pour les fonctions.*

```
#define USER_BAUDRATE  0xFFFF
#define macroFunction(x, y)      \
      do {                       \
            if (x != 10)         \
                  faireCeci(y);  \
      } while ( 1)
```

3. Commentaires

Toujours penser à expliquer ce que votre code fait et non comment il le fait !

- ***Corps fonctions :*** *Toujours utiliser ''**//**'' (C99 style). Attention, cette règle ne respecte pas le style Linux.*

- ***Cartouches fonctions ou en-têtes fichiers :*** *Toujours utiliser ''**/* ... */**'' (C89 style).*

- ***Marqueurs (Doxygen) :*** *Insérer dans vos commentaires quelques tags standards, par exemple ceux de Doxygen. Doxygen (comme JavaDoc) permet la génération automatique de documentation vers différents formats (HTML, PDF, LaTeX ...) si nous respectons quelques règles simples d'insertion de tags dans nos commentaires. Ne pas utiliser de caractères accentués dans vos commentaires.*

```
/**
 * @file demoCode.c
 * @brief demo Commentaires
 * Programme de demonstration pour la generation d'une doc Doxygen
 *
 * @author nom prenom groupe promo
 * @version 0.9
 * @date 24 decembre 2012
 */
...

int dataConv = 6;       /*donnee convertie ADC*/

...

/**
 * @struct strDataCnv
 * @brief Objet latch donnees converties
 * Sauvegarde donnee apres conversion par l'AD7075
 */
typedef struct {
      int buffSize
      int* allocCnvTab
} StrDataCnv;
```

```
...

/**
 * @fn handleSpiModule dataAcqu_AIC23 (char resetVal, float* cnvResult)
 * @brief acquisition donnee depuis codec AIC23.
 * Utilisation modules SPI_2A et SPI_3A  cote MCU PIC32MX
 * @param resetVal RAZ registre d'acquisition
 * @param  cnvResult valeur de retour apres conversion
 * @return différent de zéro si acquisition réalisee
 */
handleSpiModule dataAcqu_AIC23 (char resetVal, float cnvResult);

...

/**
 * @fn handleSpiModule dataAcqu_AIC23 (char resetVal, float* cnvResult)
 * @brief acquisition donnee depuis codec AIC23
 */
handleSpiModule dataAcqu_AIC23 (char resetVal, float cnvResult) {

      int i;

      // Algorithme de filtrage récursif
      for(i = 0; i < SIZE; i++) {
            // Corps de la boucle
      }
}
```

- *Quelques tags supplémentaires (cf. www.doxygen.org):*

```
/**
 * @example insertion d'un exemple
 * @code
 *  insertion d'une section de code (illustration d'un exemple)
 * @endcode
 * @warning insertion d'un message d'alerte
 * @li insertion d'une puce
 * @mainpage insertion de commentaires en première page de la
 * documentation Doxygen générée
 * @image  insertion d'une image
 */
```

4. Fonctions

- ***Valeur de retour :*** *Essayer de faire en sorte que la valeur de retour d'une fonction soit représentative du bon déroulement de celle-ci. Par exemple, retourne zéro (ou pointeur nul) en cas d'échec et différent de zéro en cas de succès.*